Mach Learn (2016) 102:247–273
DOI 10.1007/s10994-015-5520-1

Spatio-temporal convolution kernels

Konstantin Knauf[1] · **Daniel Memmert**[2] · **Ulf Brefeld**[3]

Received: 11 December 2014 / Accepted: 25 June 2015 / Published online: 21 July 2015

Abstract Trajectory data of simultaneously moving objects is being recorded in many different domains and applications. However, existing techniques that utilise such data often fail to capture characteristic traits or lack theoretical guarantees. We propose a novel class of spatio-temporal convolution kernels to capture similarities in multi-object scenarios. The abstract kernel is a composition of a temporal and a spatial kernel and its actual instantiations depend on the application at hand. Empirically, we compare our kernels and efficient approximations thereof to baseline techniques for clustering tasks using artificial and real world data from team sports.

Keywords Convolution kernel · Spatio-temporal · Trajectory · Soccer

1 Introduction

Trajectory data of simultaneously moving objects is the key to analyse animal migration (Lee et al. 2008), transportation (Baud et al. 2007; Giannotti et al. 2007), tactics in team sports (Hirano and Tsumoto 2005; Kempe et al. 2014; Lucey et al. 2013; Wei et al. 2013), players and

Editor: Tijl De Bie.

✉ Ulf Brefeld
brefeld@cs.tu-darmstadt.de

Konstantin Knauf
knauf.konstantin@gmail.com

Daniel Memmert
memmert@dshs-koeln.de

1 Knowledge Mining and Assessment Group, TU Darmstadt, Darmstadt, Germany

2 Institute of Cognitive and Team Sport Research, German Sport University Cologne, Cologne, Germany

3 Knowledge Mining and Assessment Group, TU Darmstadt and DIPF Frankfurt/Main, Darmstadt, Germany

avatars in (serious) computer games (Kang et al. 2013; Pao et al. 2010), customer behaviour (Larson et al. 2005) as well as spread patterns of fires (Trunfio et al. 2011). A characteristic trait of many such applications is that trajectories of several objects are more informative than the trajectory of a single object. For instance, a single trajectory of a bird is not indicative for bird migration as individuals may join or leave the flock (Lee et al. 2008) and a single trajectory of a soccer player does not reveal insights on the actual situation on the pitch (Grunz et al. 2012; Wei et al. 2013).

Therefore, trajectories of multiple objects need to be processed together. Although this insight sounds trivial, processing multiple trajectories simultaneously challenges the standard model of computation as trajectories may interdepend in time and space in multiple ways. To exploit these dependencies, it is necessary to establish a notion of similarity for spatio-temporal paths of multiple objects to identify frequent patterns. By definition, frequent patterns are formed by an *a priori* unknown subset of objects at unknown locations in time and space. Analysing multi-trajectory data is therefore inherently a combinatorial problem that involves processing data at large scales.

A second problem arises from existing methods for analysing spatio-temporal data. Traditional approaches often cannot deal with continuous spatial domains but rely on an appropriate discretisation of the data at-hand (Kang and Yong 2008; Mamoulis et al. 2004; Mehta et al. 2005). However, finding an *a priori* optimal discretisation is often difficult in many domains where only the final result allows conclusions on whether an initial set of atomic events is plausible or not (Kang and Yong 2008). Furthermore, many approaches cannot deal with permutations of the objects and differences in speed, while still being sensitive to differences in the direction of the motion (Hirano and Tsumoto 2005; Junejo et al. 2004; Wei et al. 2013).

We devise a novel class of convolution kernels for multi-trajectory data. It is specially tailored to multi-object scenarios, i.e. trajectories of multiple simultaneously moving objects. The kernel properties as well as the modular nature of the proposed class of kernels renders it highly adaptive to different applications. Since it is a kernel, it can also naturally be deployed with any kernel machine. The three characteristics, *multi-object scenario*, *modularity* and *kernel property*, distinguish our approach from existing methods. Due to its distinct characteristics, our approach is more suitable for a large variety of applications, it is flexible with respect to the notion of similarity, and it is theoretically better grounded than most of the existing methods. Since the complexity of a kernel evaluation is quadratic in the number and the lengths of the involved objects, we also propose an efficient percental approximation. Empirically, the method is evaluated on artificial datasets and real-world tracking data from ten Bundesliga soccer matches. We generally observe that our convolution kernels lead to better clusterings compared to baseline methods.

The remainder of this article is structured as follows. Section 2 reviews existing work. Section 3 introduces our spatio-temporal convolution kernel methods and Sect. 4 reports on empirical results. Section 5 concludes.

2 Related work

2.1 Trajectory clustering

Trajectory clustering, or clustering of spatio-temporal data respectively, has been an active field of research in the past years. Existing approaches mainly focus on the application of video surveillance with the goal to detect anomalies in the data stream (Basharat et al. 2008; Fu et al. 2005; Hu et al. 2006; Jeong et al. 2011; Junejo et al. 2004; Saleemi et al.

2009). Other applications include automatic sports analysis, weather evolution modelling, animal migration and traffic analysis. Existing approaches rely mostly on processing of single trajectories. Recent contributions in this area can be roughly grouped into *similarity-based approaches* (Buzan et al. 2004; Jinyang et al. 2011; Fu et al. 2005; Hirano and Tsumoto 2005; Hu et al. 2006; Junejo et al. 2004; Piciarelli et al. 2005) and *motion-based approaches* (Basharat et al. 2008; Wang et al. 2008; Jeong et al. 2011; Li and Chellappa 2010; Lin et al. 2009; Saleemi et al. 2009).

Similarity-based approaches define pairwise similarities between trajectories which are then processed by some clustering algorithm. Junejo et al. (2004) represent trajectories as a set of two-dimensional coordinates together with the Hausdorff distance. Subsequently, graph-cuts are deployed to recursively partition the trajectories. Hausdorff distances are also used to cluster trajectories by Jinyang et al. (2011) where not only the position but also the direction of the trajectories is taken into account by using 4-tuples $(x, y, \mathrm{d}x, \mathrm{d}y)$ instead of coordinates only. Fu et al. (2005) first resample trajectories to obtain constant between-point distances. Then the corresponding points of two trajectories are compared using an RBF kernel where the longer trajectory is cut to the length of the shorter one. Spectral clustering is then used together with a symmetric normalised Laplacian.

Buzan et al. (2004) extend the longest common subsequence algorithm to three-dimensional coordinates and use a modified version of agglomerative hierarchical clustering. Hirano and Tsumoto (2005) deploy multi-scale matchings to compare trajectories. The basic idea is to generate trajectories at different scales as convolutions of the trajectory and Gaussian kernels with different standard deviations. Their similarity measure is then based on the hierarchical structure of the trajectory segments at different scales. Subsequently, a rough clustering is employed. Piciarelli et al. (2005) define a trajectory-to-cluster similarity by the average Euclidean distance of trajectory coordinates to the nearest cluster coordinate where offsets in time induce negative weights.

Our approach also belongs to these similarity-based methods. In general, similarity-based approaches suffer from two major drawbacks. First, their computational complexity is at least in quadratic in the number of trajectories. Second, they rely on clustering full trajectories and are hence sensitive to tracking errors and sub-trajectories. While the first drawback is inherent to all similarity-based methods, our distribution-based approach and gradual weighting mitigates the effects of noise and tracking errors and is able to identify partial matchings between trajectories.

In contrast to similarity-based approaches, *motion-based approaches* focus on local movements of objects to derive models for the overall (group) motion in a scene. Wang et al. (2008) and Jeong et al. (2011) represent a trajectory by bags of positions as well as directions based on the bag-of-words representation of documents in natural language processing. To this end, the spatial domain is discretised and the number of occurrences of each position in a trajectory is counted. Grimson et al. also take into account temporal information by counting the occurrences of each (discretised) direction in a trajectory. The topic model Dual-HDP (Wang et al. 2008) is used to find semantic regions, which are combined to form the different trajectories. Jeong et al. use latent dirichlet allocation (Blei et al. 2003) to obtain semantic regions. To incorporate temporal information, a hidden Markov model is trained for each topic based on the sequences which are *close* to the topic. Saleemi et al. (2009) propose kernel density estimation to learn a five dimensional distribution of transitions from (x_1, y_1) to (x_2, y_2) in time t. Markov chain monte carlo (Andrieu et al. 2003) is then deployed to sample the most likely paths given the learned transition probabilities.

Basharat et al. (2008) also learn a model for transition probabilities. Instead of kernel density estimation, a Gaussian mixture model is fitted to the observed transitions. Lin et al.

(2009) exploit the Lie algebraic structure of affine transformations to learn a flow model consisting of overlapping two-dimensional Gaussian distributions, each of which corresponds to an affine transform dominant in this spatial area. The approach is applied to pedestrians in a train station and optical flows obtained from satellite images. Li and Chellappa (2010) also use a similar Lie algebraic representation called *spatial hybrid driving force model*, which, opposed to Lin et al. (2009), evolves over time. This model is used to solve the so-called group motion segmentation problem, i.e. to answer the question of which objects take part in an organised group motion and which do not.

Motion-based approaches also inhere some limitations. First, they often neglect temporal information at least of second order (curvature). Second, they do not provide a mapping of the input trajectories to groups of similar trajectories but rather describe the combined motion of all objects in all trajectories over time. Our approach differs methodologically from the summarised techniques in several ways: First, it provides a general framework that covers many applications and properties as opposed to being a very specific similarity measure tailored to a single application domain. Second, our approach is specialised on multiple simultaneously moving objects instead of focussing on only trajectories of single objects. Third, being a kernel the similarity measure is straightforwardly applicable to a broad range of algorithms and is theoretically well grounded in contrast to heuristic approaches.

2.2 Sports analytics

Current approaches in the area of sports game trajectory analysis either aim to define objective performance measures for players (Kang et al. 2006), classify (Bialkowski et al. 2013; Hervieu and Bouthemy 2010; Intille and Bobick 2001; Grunz et al. 2012; Perše et al. 2009; Siddiquie et al. 2009) or cluster (Hirano and Tsumoto 2005; Wei et al. 2013) plays/trajectories, or learn a motion model for team behaviour (Bialkowski et al. 2014; Direkoglu and O'Connor 2012; Kim et al. 2010; Li et al. 2009; Li and Chellappa 2010; Lucey et al. 2013; Zhu et al. 2007).

Kang et al. (2006) define performance metrics for soccer players based on the definition of owned and competitive regions of the field, which are derived from ball and player trajectories. Siddiquie et al. (2009) represent videos of football plays by a bag-of-features from histograms of optical flows as well as histograms of oriented gradients. Spatio-temporal pyramid matching (Lazebnik et al. 2006) is used to generate a kernel for each visual word. Football plays are then classified into seven categories using multiple kernel learning. Hervieu and Bouthemy (2010) use a hierarchical parallel semi-Markov model to classify different activity states in squash and handball, such as rallies, free throws and defence. The first layer describes the activity states, while the second layer consists of a parallel hidden Markov model for each feature representing the trajectories.

Perše et al. (2009) represent team activity in basketball using team centroids to hierarchically classify situations with Gaussian Mixture Models. Thereafter, each situation is converted into a string, which is compared to templates for classification. Bialkowski et al. (2013) use team centroids and occupancy maps to classify game situations in field hockey (corners, goals), emphasising the robustness of this representation to tracking noise. Grunz et al. (2012) employ self-organising maps to identify long and short game initiations in soccer and Hirano and Tsumoto (2005) use multi-scale comparison and rough clustering to cluster ball trajectories that lead to goals.

Direkoglu and O'Connor (2012) solve a special Poisson equation, in which the player positions determine the location of source terms. The derived distribution and its development over time defines a so-called region of interest used to describe the team behaviour. Wei et al. (2013) use role models (Lucey et al. 2013) and a Bilinear spatio-temporal basis model to

represent team movement to cluster goal scoring opportunities in soccer. Bialkowski et al. (2014) also use role models to automatically detect and compare the formations of soccer teams. Li and Chellappa (2010) learn a spatio-temporal driving force model to identify offence and defence players in football. Kim et al. (2010) interpolate a dense motion field from player trajectories using thin-plate splines. This motion field is further investigated for points of convergence to predict where the game will evolve in short term.

From an application point of view, our approach is most comparable to Wei et al. (2013) and Grunz et al. (2012). While Wei et al. focus on scoring opportunities and Grunz et al. study game initiations, we consider both situations in this study. Similar to Bialkowski et al. (2013), our method proves robust to tracking noise.

3 Spatio-temporal convolution kernels

3.1 Representation

Multi-object trajectory analysis is concerned with a possibly varying number of moving objects $\mathcal{O}_t$ in a set X, e.g. $X = \mathbb{R}^2$, over a finite period of time $\mathcal{T} \subset \mathbb{N}$. A multi-object trajectory is composed of snapshots of the object positions at different times. Depending on the context and application at hand, one of the following two formalisations of a snapshot is more appropriate.

Definition 1 (Object-oriented Snapshot) Assume the number of objects to be constant over time, i.e. $\mathcal{O}_t = \mathcal{O} = \{o_1, \ldots, o_N\}$ for $N \in \mathbb{N}$. Then the object-oriented snapshot of all objects at time $t \in \mathcal{T}$ is denoted by $x_t \in X^N =: \mathcal{X}$. We call $\mathcal{X}$ the snapshot space. The position of a particular object $o \in \mathcal{O}$ is denoted by $x_t(o) \in X$.

Definition 2 (Group-oriented snapshot) Assume there is a constant number of groups $K \in \mathbb{N}$. Moreover, at every point in time each object can be associated with exactly one of the groups $\mathcal{G} = \{g_1, \ldots, g_K\}$.[1] Then the group-oriented snapshot of all objects at time $t \in \mathcal{T}$ is denoted by

$$x_t \in \mathcal{P}(X)^K =: \mathcal{X}.$$

We call $\mathcal{X}$ the snapshot space. The positions of all objects of a particular group $g \in \mathcal{G}$ are denoted by $x_t(g) \in \mathcal{P}(X)$. The group members of group g in snapshot x_t are denoted by $O_{x_t}(g) \subset \mathcal{O}_t$.

The implications of the two definitions are as follows. First, the object-oriented snapshot representation only allows a fixed number of objects, whereas the group-oriented representation is not limited in that respect. Second, in the group-oriented snapshot, objects inside a group are indistinguishable. On one hand, the property allows for permutations of objects but on the other hand it naturally also entails ambiguities.

Instead of an ordered sequence of positions or snapshots we use a set of time/position-pairs to represent trajectories. Thereby, time and order is explicitly represented as opposed to the more implicit sequence representation.

Definition 3 (Trajectory) A trajectory is defined as a finite subset

$$\tilde{P} = \{(\tilde{t}_1, x_{\tilde{t}_1}), \ldots, (\tilde{t}_n, x_{\tilde{t}_n})\} \subset \mathcal{T} \times \mathcal{X},$$

[1] Note that the group membership of an object can change over time.

such that $\tilde{t}_i \neq \tilde{t}_j$ for $i \neq j$, i.e. the trajectory set $\tilde{P}$ contains only one snapshot per point in time.

The set $\pi_{\mathcal{T}}(\tilde{P})) = \{t \in \mathcal{T} : \exists(\tilde{s}, x_{\tilde{s}}) \in \tilde{P} \text{ s. t. } t = \tilde{s}\}$ contains all timestamps of the trajectory and is usually of the form $\{K, K+1, \ldots, K+L\}$ for some natural numbers K and L. When comparing trajectories it is insignificant at what absolute time the trajectories start. This gives rise to the following definition.

Definition 4 (Time-normalised trajectory) The time-normalised trajectory $P \subset [0, 1] \times \mathcal{X}$ corresponding to trajectory $\tilde{P}$ is defined by normalising its time-scale to $[0, 1]$. This corresponds to the trajectory P, given by

$$P = \{(t, x_t) : \exists(\tilde{t}, x_{\tilde{t}}) \in \tilde{P} \text{ s.t. } \tilde{t} = \mu + t(\max(\pi_{\mathcal{T}}(\tilde{P})) - \mu) \wedge x_{\tilde{t}} = x_t\},$$

where $\mu = \min(\pi_{\mathcal{T}}(\tilde{P}))$.

In the remaining part of this study we refer to time-normalised trajectories simply by trajectories.

3.2 Problem setting

One of the main advantages of kernel methods is the separation of algorithm and data. Following this paradigm, we focus on defining a kernel on the set of multi-object trajectories. Once a kernel has been defined, off-the-shelf kernel machines can be applied to generate models, such as support vector machines (Vapnik 1995), kernelised k-medoids (Kaufman and Rousseeuw 1987), or spectral clustering (Ng et al. 2001). The formal problem setting of this article is defined as follows. On the set of multi-object trajectories $M \subset \mathcal{P}([0, 1] \times \mathcal{X})$ we aim to develop a similarity measure $k : \mathcal{P}([0, 1] \times \mathcal{X}) \times \mathcal{P}([0, 1] \times \mathcal{X})$, such that

(I) the absolute position as well as the shape of the trajectories is incorporated,
(II) the measure is invariant to permutations of certain objects, i.e. for two trajectories P_1, P_2 with

$$P_2 = \{(t, x_t) : \exists \text{ permutation } \sigma \; \forall(s, y_s) \in P_1 \text{ s. t. } t = s \wedge x_t = \sigma(y_s)\}$$

it holds that $k(P_1, P_2) = 1$. In case of the group-oriented snapshot this already holds by definition if the permuted objects are members of the same group,
(III) the measure is invariant with respect to the speed of the movement. Since all trajectories have already been normalised to the same time scale, differences in speed are mainly reflected in the cardinality of the trajectory sets. So, for example, given two trajectories P_1 and P_2 with $|P_1| = 2|P_2|$ and

$$P_2 = \{(t, x_t) : \exists(s, y_s) \in P_1 \text{ s. t. } t = 2s \wedge x_t = y_{2s}\}$$

it holds that $k(P_1, P_2) \approx k(P_1, P_1)$,[2]
(IV) the similar movements of two sets of objects is recognised as such in the presence of deviations of single objects and outliers.

[2] Note that the Definition of P_2 is such that it corresponds to an object moving with twice the speed of the first object.

Moreover, the measure should have the following properties:

(V) Kernel Property, i.e.

$$k(P_1, P_2) = \langle \phi(P_1), \phi(P_2) \rangle_{\mathcal{F}}$$

for some, usually unknown, feature map ϕ and inner product space $\mathcal{F}$

(VI) Broad applicability, i.e. few application specific parameters and no restrictions on the space X

(VII) Computational efficiency

Note that properties (I) to (IV) formalise an intuitive notion of similarity. (I) says that shape and positions matters. (II) requires that if two similar object swap roles, it does not matter. (III) formalises that we do not care about differences in speed that much.[3]

Property (IV) demands robustness with respect to outlier trajectories. Further note that, for example, Dynamic Time Warping (Bellman and Kalaba 1959) meets condition (III) very well, but does not comply with (V), (VI) and (VII), since it is not a kernel and only applicable if the underlying set is a metric space. Moreover, it is computationally expensive. On the other hand, the Hausdorff distance (Hausdorff 1962) satisfies (I), (III) and (VII), but it does not satisfy (II), (V) and (VI), since it is only applicable to metric spaces and sensitive to permutations. In addition, it is not a kernel. A Gaussian RBF kernel on the full vector of positions meets conditions (I) (restricted), (V) and (VII), but is not applicable to sequences of different lengths. Also, it does not comply with (II),(III) and (VI), since it is highly sensitive to variations in speed and permutations and is restricted to metric spaces.

3.3 Spatio-temporal convolution kernels for multi-trajectories

In this section we develop a kernel on the space of (time-normalised) multi-trajectories $\mathcal{P}([0, 1] \times \mathcal{X})$. Each of those trajectories consists of a set of snapshots associated with a relative time. The general idea is to perform a pairwise comparison of the snapshots in the two sets. Therefore, we first need a way to compare snapshots and, second, we need to know which snapshots of the two trajectory sets to compare with each other. For the latter dynamic time warping (DTW) (Bellman and Kalaba 1959) seems to be a good choice, since it aligns the snapshots optimally in terms of similarity. Unfortunately, the obtained kernel is not positive definite, i.e. it does not correspond to an inner product in some Hilbert Space. Although there is anecdotal evidence that learning with indefinite kernels can lead to good results in some applications (e.g. Ong et al. 2004), theory only supports the use of positive definite kernels. For many kernel machines there are error bounds and convergence criteria that can be straightforwardly applied to positive definite kernels but that do not hold for indefinite kernels (Blanchard et al. 2008; Lin 2001; Steinwart 2002).

Therefore, we propose a weighted comparison between every snapshot of the first trajectory and every snapshot of the second one where the weights depend on the offset in relative time. Formally, this is done using an R-convolution kernel (Haussler 1999) on the two sets representing the trajectories. Convolution kernels are a general class of kernels on structured objects $x, y \in X$.[4] The idea is to compare instances x and y by comparing their *parts* $(x_1, \ldots, x_D), (y_1, \ldots, y_D) \in X_1 \times \cdots \times X_D$. Thus, a relation function R is needed to express that something is a *part* of some structure.

[3] Note that the property can be easily adapted to explicitly include speed by adding an extra coordinate. The position of each object is replaced by a position-speed-pair $\left(x, \frac{\mathrm{d}x}{\mathrm{d}t}\right)_t$, see Jinyang et al. (2011) for details.

[4] Originally, convolution kernels are defined on arbitrary sets X.

Definition 5 (Relation) Let $X, X_1, \ldots, X_D$ be arbitrary sets. Then a relation

$$R : X \times X_1 \times \cdots \times X_D \to \{0, 1\}$$

is an arbitrary boolean function that returns 1 if and only if $(x_1, \ldots x_d)$ are parts of x. The set of parts of $x \in X$ under relation R is denoted by

$$R^{-1}(x) = \{(x_1, \ldots, x_d) \in X_1 \times \cdots \times X_D : R(x, x_1, \ldots, x_D) = 1\}$$

and R is called **finite** if $R^{-1}(x)$ is finite for every $x \in X$.

Definition 6 (R-convolution kernel) Let $X, X_1, \ldots, X_D$ be arbitrary sets. Let $x, y \in X$ and $R : X \times X_1 \times \cdots \times X_D \to \{0, 1\}$ be a finite relation. Moreover, let $k_1, \ldots, k_D$ be kernels on $X_1, \ldots, X_D$. Then the R-convolution kernel on X is defined by

$$k(x, y) = \sum_{\substack{(x_1,\ldots,x_D)\in R^{-1}(x),\\ (y_1,\ldots,y_D)\in R^{-1}(y)}} \prod_{d=1}^{D} k_d(x_d, y_d)$$

The following theorem shows that an R-convolution kernel is indeed a (positive-definite) kernel.

Theorem 1 *Let $X, X_1, \ldots, X_D$ be arbitrary sets. Let R be a finite relation and let $k_1, \ldots, k_D$ be kernels on $X_1, \ldots, X_D$. Then the R-convolution kernel k given by Definition 6 is a kernel.*

Proof For the proof we refer to Haussler (1999) Theorem 1 and Lemma 1, which are essentially more involved applications of closure properties of kernels. □

In our case the structure is a multi-object trajectory and its *parts* are the snapshots, the times of the snapshots and the length of the trajectory. The relation between the structure and its elementary components is given by

$$R : \underbrace{\mathbb{N}}_{\text{Length of Traj.}} \times \underbrace{[0, 1]}_{\text{Time of Snapshot}} \times \underbrace{\mathcal{X}}_{\text{Snapshot}} \times \underbrace{\mathcal{P}([0, 1] \times \mathcal{X})}_{\text{Trajectory}} \to \{0, 1\}.$$

To extract the snapshots with their associated times and the length of the trajectory from the trajectory, R needs to be defined as follows:

$$R(n, t, x, P) = \begin{cases} 1 & \text{if } |P| = n \wedge \exists (s, y_s) \in P : (t, x) = (s, y_s) \\ 0 & \text{otherwise.} \end{cases} \tag{1}$$

The first condition in Eq. 1 ensures that the given trajectory has the given length, while the second condition guarantees the occurrence of the given snapshot at the given time in the given trajectory. With this relation and the kernels to compare the *parts* the resulting convolution kernel is given by

$$k(P, Q) = \sum_{\substack{(n,t,x)\in R^{-1}(P),\\ (m,s,y)\in R^{-1}(Q)}} k_{\mathbb{N}}(n, m) \cdot k_{[0,1]}(t, s) \cdot k_{\mathcal{X}}(x, y) \tag{2}$$

with $R^{-1}(P) = \{(n, t, x) : R(n, t, x, P) = 1\}$. The term $k_{\mathbb{N}}$ accounts for differences in length of the trajectories by normalisation, i.e. $k_{\mathbb{N}} = \frac{1}{nm}$. Note that $k_{\mathbb{N}}$ is a kernel on $\mathbb{N}$, since

it corresponds to the standard inner product under the feature map $\phi : \mathbb{N} \to \mathbb{N}$ given by $\phi : n \mapsto \frac{1}{n}$. Finally, the R-convolution kernel simplifies to

$$k(P, Q) = \frac{1}{|P||Q|} \sum_{(t,x_t)\in P,(s,y_s)\in Q} k_{[0,1]}(t, s) \cdot k_{\mathcal{X}}(x_t, y_s). \tag{3}$$

Theorem 2 *The spatio-temporal convolution kernel (Eq. 3) is a kernel if the temporal kernel $k_{[0,1]}$ and the spatial kernel $k_{\mathcal{X}}$ are kernels in that sense.*

Proof By Theorem 1 we need to show that R is finite and the component kernels $k_{\mathbb{N}}$ and $k_{[0,1]}$, $k_{\mathcal{X}}$ are indeed kernels. First, for all $P \in \mathcal{P}([0, 1] \times \mathcal{X})$ it holds that $|R^{-1}(P)| = |P| < \infty$ by Definition 3, so R is finite. Second, $k_{[0,1]}$ and $k_{\mathcal{X}}$ are kernels by assumption and we have just shown that $k_{\mathbb{N}}$ is also a kernel. Hence, the spatio-temporal convolution kernel is a kernel. □

As indicated, Eq. 3 can be interpreted such that all snapshots of the two trajectories are compared with each other, but weighted by their offset in time. Thereby snapshots which occur at different relative times have a low contribution to the overall similarity, while snapshots at the same relative time have a high contribution. This is why $k_{[0,1]}$ is sometimes also referred to as weight function. The definition of k in Eq. 3 leaves two degrees of freedom:

- Spatial kernel $k_{\mathcal{X}}$: The choice of the snapshot kernel determines which snapshots are similar.
- Temporal kernel $k_{[0,1]}$: The choice of the temporal kernel determines the way in which the snapshots of two sequences are combined, and thus the importance of ordering and speed.

In the following subsections we develop and compare different spatial and temporal kernels.

3.4 Spatial kernels

A spatial kernel compares two snapshots in X. Corresponding to the two definitions of the snapshot in Definition 1 (object-oriented) and Definition 2 (group-oriented), two types of kernels are introduced here as well.

Object-Wise Comparison These kernels correspond to the object-oriented snapshot by simply comparing the positions of the objects $\mathcal{O}$ and summing up their similarity:

$$k_{\mathcal{X}}(x_t, y_t) = \frac{1}{|\mathcal{O}|} \sum_{o\in\mathcal{O}} k_X(x_t(o), y_t(o)). \tag{4}$$

Note that Eq. 4 is a kernel, since kernels are closed under direct sum and multiplication by a positive constant (see Shawe-Taylor and Cristianini (2004) Proposition 3.22).[5] Since kernels are also closed under direct product, technically a product could have been used instead of the sum in Eq. 4. However, a product of kernels leads to vanishing similarities if only one object is dissimilar to its counterpart. This is counterintuitive as two snapshots are the more similar the more objects match well. It is thus an inherently additive relationship. For the same reason a product of kernels is more vulnerable to noise and outliers and generally less robust than a sum of kernels. The elementary kernel k_X can be any kernel comparing two positions in X. Table 1 lists common kernels for finite as well as continuous snapshot spaces.

[5] However, if the number of objects was not constant, $k_{\mathcal{X}}$ would not be a kernel.

Table 1 Elementary spatial kernels

Name	X	Kernel	Para.
Uniform	$\mathbb{R}^n$	$k_X(x, y) = \mathbb{I}_{\{z\in\mathbb{R}^n:\|z-x\|_2<w\}}(y)$	w
Triangular	$\mathbb{R}^n$	$k_X(x, y) = \left(1 - \frac{\|x-y\|_2}{w}\right) \mathbb{I}_{\{z\in\mathbb{R}^n:\|z-x\|_2<w\}}(y)$	w
Polynomial	$\mathbb{R}^n$	$k_X(x, y) = (\langle x, y\rangle + R)^d$	R,d
Gaussian	$\mathbb{R}^n$	$k_X(x, y) = \exp\left(-\frac{1}{2\sigma^2}\|x - y\|_2^2\right)$	σ^2
Matching kernel	Finite X	$k_X(x, y) = \mathbb{I}_{\{x\}}(y)$	–

Kernels as in Eq. 4 have two major shortcomings. First, they penalise permutations of objects. For example, two snapshots of two objects with swapped positions will have low similarity, although in terms of the group motion they are alike. One way to address permutations is to explicitly maximise the similarity of the two snapshots with respect to all possible permutations of objects. Due to the high computational costs of considering all possible permutations[6], this is infeasible. In addition, the bandwidth parameter σ that controls the interval of high sensitivity, i.e. large values of $|\mathrm{d}k/\mathrm{d}(\|x - y\|)|$, of the kernel has to be known beforehand. This is critical since one usually does not know on which scale significant deviations appear. Both issues are addressed by the group-wise comparison of objects.

Group-Wise Comparison In case of the group-oriented snapshot there is a partition of $\mathcal{O}_t$ into K sets $O_{x_t}(g_1)$, $O_{x_t}(g_2)$, .., $O_{x_t}(g_K)$. Instead of comparing N objects, K groups of objects are compared, which leads to the following definition of a spatial kernel

$$k_{\mathcal{X}}(x_t, y_t) = \frac{1}{K}\sum_{g\in\mathcal{G}} k_G(x_t(g), y_t(g)) \tag{5}$$

with kernel k_G applied on sets of positions. For the same reasons as in the previous paragraph, a sum of kernels is preferable to a product of kernels. Definition 2 allows for zero group members in a snapshot, i.e. $x_t(g) = \emptyset$. For all kernels defined below this special case is dealt with as follows:

$$k_G(x, y) = \begin{cases} 1 & \text{if } x = y = \emptyset \\ 0 & otherwise. \end{cases}$$

The definition of k_G is very much dependent on the underlying set of positions X. There are two main classes of sets X. First, if X is a metric space, i.e. there exists a distance measure between all positions. Second, X is a finite set, in which no meaningful distance between positions can be defined.[7] In the case of a metric space we will subsequently only consider $X = \mathbb{R}^2$ with standard Euclidean distance as it is the most common in applications and can be directly extended to $X = \mathbb{R}^N$.

Euclidean Space: $X = \mathbb{R}^2$ In the case of $X = \mathbb{R}^2$ a straightforward approach is to use one of the kernels from Table 1 to compare the group centroids

[6] For example, 10 objects lead to about $3.6 \cdot 10^6$ permutations.

[7] The case of an infinite set X without a metric can be reduced to the finite case by only considering the positions that are attained by some objects at some time. These are only finitely many.

$$\mu_{x_t}(g) = \frac{1}{|O_{x_t}(g)|} \sum_{o \in O_{x_t}(g)} x_t(o) \quad \text{and} \quad \mu_{y_t}(g) = \frac{1}{|O_{y_t}(g)|} \sum_{o \in O_{y_t}(g)} y_t(o),$$

which leads to

$$k_G(x_t(g), y_t(g)) = k_X(\mu_{x_t}(g), \mu_{y_t}(g)).$$

Besides the need of defining the width parameter σ, disregarding the distribution of the objects around their centroids constitutes a disadvantage. To remedy both deficiencies, two Gaussian distributions are fitted to $x_t(g)$, and $y_t(g)$ respectively that we will denote as $f_{x_t}(g)$ and $f_{y_t}(g)$. These distributions are defined by their means $\mu_{x_t}(g)$ and $\mu_{y_t}(g)$ as defined above and their covariance matrices defined by

$$\Sigma_{x_t}(g) = \frac{1}{|O_{x_t}(g)| - 1} \sum_{o \in O_{x_t}(g)} (x_t(o) - \mu_{x_t}(g))(x_t(o) - \mu_{x_t}(g))^T$$

$$\text{and } \Sigma_{y_t}(g) = \frac{1}{|O_{y_t}(g)| - 1} \sum_{o \in O_{y_t}(g)} (y_t(o) - \mu_{y_t}(g))(y_t(o) - \mu_{y_t}(g))^T.$$

If the covariance matrix is ill-conditioned or singular[8], a simple shrinking scheme with shrinkage parameter α can be applied to achieve non-singularity:

$$\Sigma = (1 - \alpha) \cdot \Sigma + \alpha \cdot \frac{\mathrm{Tr}(\Sigma)}{2} \mathbb{I}_2,$$

where $\mathbb{I}_2$ is the two-dimensional identity matrix. There exist different strategies (Ledoit and Wolf 2004; Chen et al. 2010) to choose an optimal value for α, but for our purposes it suffices to deploy a constant 0.1. In the case of $\mathrm{Tr}(\Sigma) = 0$ the following scheme is used

$$\Sigma = (1 - \alpha) \cdot \Sigma + \alpha \cdot \sigma^2_{\mathrm{MIN}} \cdot \mathbb{I}_2 = \alpha \cdot \sigma^2_{\mathrm{MIN}} \cdot \mathbb{I}_2,$$

where σ^2_{MIN} is an application specific parameter. Strategies for choosing σ^2_{MIN} are discussed in Sect. 4. Reasons for the matrix to be singular are $O_t(g) \leq 2$ or strong collinearity of the samples. The two Gaussian distributions $f_{x_t}(g)$ and $f_{y_t}(g)$ are then compared using a probability product kernel (Jebara et al. 2004), i.e.

$$k_G(x_t(g), y_t(g)) = k^{\mathrm{prod}}(f_{x_t}(g), f_{y_t}(g)).$$

A probability product kernel compares two probability distributions, i.e. their density functions p and p', defined on a common probability space $(\Omega, \mathcal{A}, \mu)$. It is defined as follows.

Definition 7 (Probability Product Kernel) Let p and p' be probability density functions defined on the same probability space $(\Omega, \mathcal{A}, \mu)$. Assume that $p^\rho, p'^\rho \in L^2(\Omega)$ for $\rho \in \mathbb{R}^+$. Then the probability product kernel (PPK) $k^{\mathrm{prod}} : L^2(\Omega) \times L^2(\Omega) \to \mathbb{R}$ is given by

$$k^{\mathrm{prod}}(p, p') = \int_\Omega p(\omega)^\rho p'(\omega)^\rho \, \mathrm{d}\mu(\omega).$$

Lemma 1 *k^{prod} is a (positive-definite) kernel.*

[8] In this study a threshold on the covariance matrix's condition number of 30 is used.

Proof Consider the feature map $\phi : L^2(\Omega) \to L^2(\Omega)$ given by

$$\phi(p) = p^\rho,$$

which is well-defined because of above L^2 property. Then

$$k^{\text{prod}}(p, p') = \langle \phi(p), \phi(p') \rangle_{L^2(\Omega)}$$

and k^{prod} is a kernel corresponding to the scalar product in $L^2(\Omega)$

$$\int_\Omega f(\omega) \cdot g(\omega) \, \mathrm{d}\mu(\omega).$$

□

In this study we will use $\rho = 1/2$, which has the important property that $\text{Im}\,(k_{Prod}) = [0, 1]$ and $k^{\text{prod}}(p, p) = 1$. For two two-dimensional Gaussian distributions and $\rho = 1/2$ the product probability kernel permits the following closed form:

$$\begin{aligned} k_G(x_t, y_t) = \int_{\mathbb{R}^2} \left(f_{x_t}(z) f_{y_t}(z)\right)^{1/2} \mathrm{d}z = 2 \cdot |\Sigma^*|^{\frac{1}{2}} |\Sigma_{x_t}|^{-\frac{1}{4}} |\Sigma_{y_t}|^{-\frac{1}{4}} \\ \cdot \exp\left(-\frac{1}{4}\left(\mu_{x_t}^T \Sigma_{x_t}^{-1} \mu_{x_t} + \mu_{y_t}^T \Sigma_{y_t}^{-1} \mu_{y_t} - \mu^{*T} \Sigma^* \mu^*\right)\right) \end{aligned} \tag{6}$$

with $\Sigma^* = (\Sigma_{x_t}^{-1} + \Sigma_{y_t}^{-1})^{-1}$ and $\mu^* = \Sigma_{x_t}^{-1} \mu_{x_t} + \Sigma_{y_t}^{-1} \mu_{y_t}$.[9] Besides its sensibility to the distribution of the objects inside the group k^{prod} has the advantage of not having to choose a width parameter. The kernel essentially adapts to the scale of the group.

Finite Case: $|X| = L < \infty$ For finite $X = \{z_1, \ldots, z_L\}$, we can generalise to the group-oriented snapshot by comparison of the number of objects of each group at each possible location in X:

$$k_G(x_t(g), y_t(g)) = \frac{1}{L} \sum_{z \in X} \mathbb{I}_{\{n_z(x_t(g))\}}(n_z(y_t(g))) \tag{7}$$

with the indicator function $\mathbb{I}_A(x) = 1$ if $x \in A$ and 0 otherwise and $n_z(x_t(g)) = |\{x_t(o) = z \quad o \in O_{x_t}(g)\}|$ denotes the number of objects of group $g \in \mathcal{G}$ in position $z \in X$ at time $t \in [0, 1]$.

Lemma 2 *k_G as defined in Eq. 7 is a kernel.*

Proof Let $l^2(0, 1)$ be the Hilbert space of 0/1-sequences equipped with the standard scalar product. Consider $\phi : X \to l^2(0, 1)^L$ with

$$\phi(x_t(g))_{m,l} = \begin{cases} 1 & \text{if m objects in position l} \\ 0 & \text{otherwise} \end{cases}$$

for $l = 1, \ldots, L$ and $m = 0, 1, 2, \ldots$ Using this feature mapping, k_G can be rewritten to

$$k_G(x_t(g), y_t(g)) = \langle \phi(x_t(g)), \phi(y_t(g)) \rangle,$$

showing that it is indeed a kernel. Note that that the inner product of a product space is defined by the sum of the inner products of its components. □

[9] In order to simplify the notation, group index g has been omitted.

If the number of possible locations is much higher than the number of objects, the above kernel (Eq. 7) is inappropriate, as it will return similarities close to one for every pair of snapshots because in both snapshots there are no objects at almost all locations. Alternatively, the idea of a product probability kernel can also be applied to the finite setting by fitting a multinomial distribution to the positions of the group and comparing the two resulting distributions $p_{x_t}(g)$ and $p_{y_t}(g)$ that are identified by their outcome probabilities $(p_{x_t,1}(g), \ldots, p_{x_t,L}(g))$. For $\rho = 1/2$ the probability product kernel on the multinomial distributions is derived as follows[10],

$$\begin{aligned} k^{\text{prod}}(x_t, y_t) &= \int_{\mathbb{N}^L} \sqrt{p_{x_t}(n_1, \ldots, n_L) p_{y_t}(n_1, \ldots, n_L)}\, \mathrm{d}(n_1, \ldots, n_L) \\ &= \sum_{n_1 \in N} \cdots \sum_{n_L \in N} \sqrt{p_{x_t}(n_1, \ldots, n_L) p_{y_t}(n_1, \ldots, n_L)} \\ &= \sum_{n_1 \in N} \cdots \sum_{n_L \in N} \binom{n_1 + \cdots + n_L}{n_1, \ldots, n_L} \left((p_{x_t,1} \cdot p_{y_t,1})^{n_1} \cdot \cdots \cdot (p_{x_t,L} \cdot p_{y_t,L})^{n_L} \right)^{\frac{1}{2}} \end{aligned}$$

with the maximum-likelihood estimation for the outcome probabilities of the multinomial distributions given by

$$p_{x_t,l}(g) = \frac{n_{z_l}(x_t(g))}{|O_{x_t}(g)|}.$$

3.5 Temporal kernels

The temporal kernel $k_{[0,1]}$ is the simpler component of spatio-temporal convolution kernel because the underlying set is fixed to the one-dimensional interval [0, 1]. We briefly discuss possible options for the temporal kernel and their implications.

The simplest temporal kernel is a constant kernel $k_{[0,1]}(t, s) = 1$ with the consequence that the spatio-temporal convolution kernel (Eq. 3) collapses to a set kernel on the two sets of snapshots, thus ignoring order. A uniform or interval kernel given by

$$k_{[0,1]}(t, s) = \mathbb{I}_{\{u \in \mathbb{R} : |t-u| < w\}}(s)$$

could be used to compare every snapshot of the first trajectory to just those snapshots of the second trajectory which are close in (relative) time. The choice of w determines how close snapshots have to be to be taken into consideration. Finally, Gaussian kernels of the form

$$k_{[0,1]}(t, s) = \exp\left(-\frac{1}{2\sigma^2} \|t - s\|_2^2 \right)$$

compare every snapshot of the first trajectory to every snapshot of the second trajectory and gives more importance to closer events (in relative time). Depending on the application at-hand, there may be many more options to define temporal kernels.

3.6 Approximation techniques

To compute the Gram matrix of a dataset of N trajectories, $O(N^2 L^2)$ spatial as well as temporal kernels need to be evaluated with L being the maximal length of a sequence. Naturally, the evaluation of the spatial kernels will dominate the evaluation of the temporal kernel mainly for reasons of dimensionality as well as complexity of the kernel itself.

[10] The group index g has again been omitted for reasons of readability.

Algorithm 1 ASTCK: Percental Approximation

```
1: function ASTCKII(P,Q,L)   ▷ Input: Two Trajectories P and Q and the max. length of any trajectory L,
   Output: Vector of step-by-step approximation of k(P,Q)
2:    tempMatrix = GETTEMPORALKERNELMATRIX(P,Q);
3:    orderedEntries = SORT(tempMatrix)
4:    k(0) = 0;
5:    for  i=1 to L do
6:       k(i) = k(i-1)·ROUND((i-1)/L·|P|·|Q|);
7:       for j = round((i-1)/L·|P|·|Q|)+1 to round(i/L·|P|·|Q|) do
8:          k(i) += SPATIALKERNEL(P(orderedEntries(j,1)) Q(orderedEntries(j,2)));
9:       end for
10:       k(i) = k(i)/ROUND(i/L·|P|·|Q|)
11:   end for
12:   return k;
13: end function
```

Generally, there exist two ways to speed up the algorithm. First, the number of similarities which have to be computed is reduced to shrink the factor $O(N^2)$. Afterwards the missing entries in the Gram matrix are reconstructed. Second, the computation of each similarity is accelerated to reduce the term $O(L^2)$. Regarding the first approach, there exist methods to reconstruct incomplete Gram matrices, most notably the Nyström method (see Fowlkes et al. 2004, Williams and Seeger 2001). These methods have the disadvantage of usually needing a constant fraction of all entries to perform well (Wauthier et al. 2012). It follows that they do not improve the asymptotic complexity of the computation. We focus on the second approach, since it is specific to spatio-temporal convolution kernels and potentially improves the asymptotic complexity.

As stated before, the computation of the temporal kernel is significantly faster than the one of the spatial kernel. Thus, a natural way to approximate these kernels is to first evaluate all temporal kernels and then to only evaluate those spatial kernels which have the highest contribution. Depending on the lengths of the two trajectories at-hand the number of overall kernel evaluations varies between pairs of trajectories. Therefore, when approximating the kernel one has to take care that all entries of the Gram matrix are approximated in similar speed to avoid distortions. To this end, we propose a percental approximation algorithm (Algorithm 1, Approximated STCK (ASTCK)), which in addition to the two trajectories takes the maximal length of all trajectories L as an input. The algorithm first evaluates all temporal kernels. Subsequently, in each step $\frac{1}{L}$ spatial kernels are computed starting with the most relevant based on the evaluation of the temporal kernels. The order of evaluation is illustrated in Table 2. This algorithm leads to a complexity of

$$\mathcal{O}(N^2 \cdot L^2 \cdot C_T + N^2 \cdot L \cdot I \cdot C_S),$$

where C_T is the complexity of evaluating the temporal kernel, C_S is the complexity of evaluating the spatial kernel and I is the number of iterations of the approximation algorithm. The parameter $1 \leq I \leq L$ needs to be chosen by the user. In practice, small values of I often lead to highly accurate results.

3.7 Online application

STCKs are particularly well-suited for online analyses such as real-time computations where the objects of interest are still in motion. When a new measurement is added to a trajectory, Eq. 3 can be efficiently updated, since all previous spatial kernel evaluations remain constant

Table 2 Evaluation order of ASTCK with $L = 12$ and $|P| = |Q| = 6$

		Timesteps P					
	Step	0	.2	.4	.6	.8	1
Timesteps Q	0	1	3	6	9	11	12
	.2	4	1	3	6	9	11
	.4	7	5	1	3	7	9
	.6	10	8	5	2	4	7
	.8	11	10	8	5	2	4
	1	12	12	10	8	6	2

in the sum. That is, only the value of the temporal kernel needs to be computed which is however usually inexpensive.

4 Empirical evaluation

In this section we empirically compare our spatio-temporal convolution kernels to baseline approaches using artificial and real data sets. We focus on clustering tasks and k-medoids (Kaufman and Rousseeuw 1987) as the underlying learning algorithm. The temporal kernel is always a Gaussian kernel that we combine with three different spatial kernels: an object-wise Gaussian RBF kernel as spatial kernel ($\text{STCK}_{\text{1-1}}$), a Gaussian RBF kernel on the group means ($\text{STCK}_{\text{Mean}}$), and a probability product kernel on the fitted Gaussian distributions ($\text{STCK}_{\text{Dist}}$).

For the latter, the group and application specific parameter σ^2_{MIN} which is sometimes needed to restore non-singularities of the covariance matrix (see Sect. 3.4), is set to the average distance between two objects of the group. If the group only consists of one object, it is equal to the average distance between two objects of the group, that is most similar. The width parameter σ_T of the temporal Gaussian kernel is set to 0.5 to balance invariance to speed differences and sensitivity to direction. The width parameter σ_S of the spatial Gaussian RBF kernel is set to 0.2.

We deploy three baselines. First, Junejo et al. (2004) is straightforwardly extended to the multi-object scenario, i.e. the use of Hausdorff distance on the set of positions of the trajectories. Instead of the hierarchical clustering employed in Fu et al. (2005), kernelised k-medoids is used. The second baseline is inspired by Wang et al. (2008). We use a bag-of-positions as well as a bag-of-directions representation for the trajectories of each group. To keep the setup simple, we use a multinomial mixture model (*MNMM*) and expectation maximisation for clustering instead of a semantic topic model like dual-HDP (Wang et al. 2008). Third, we also compare our method to dynamic time warping with a product probability kernel (DTW_{dist}) serving as local distance measure. This method applies the product probability kernel to the fitted Gaussian distributions. The number of clusters is determined using the silhouette measure (Rousseeuw 1987), Hartigan index (Hartigan 1975) as well as next-neighbour consistency for all methods.

In the next section, we measure the alignment between predicted and ground-truth clusterings using artificially generated data. The alignment between two groupings is captured by the Rand Index (Rand 1971); however, as two random clusterings may return a non-zero Rand Index by chance, we resort to the Adjusted Rand Index (Hubert and Arabie 1985), given by

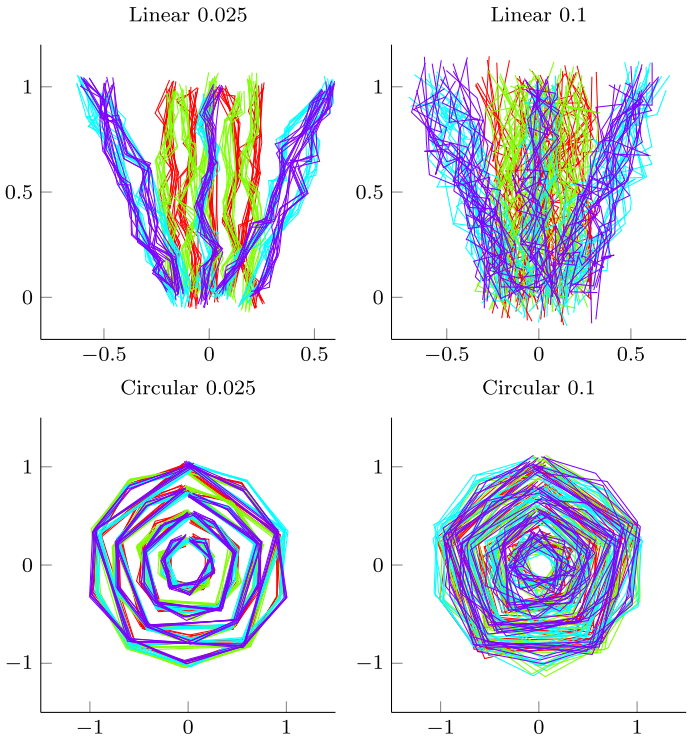

Fig. 1 Exemplary artificial multi-trajectories. For both settings ten multi-trajectories for each cluster are shown. *Colours* indicate cluster membership. Moreover, two levels of noise are depicted. For instance, *linear 0.025* show linear movements with added noise in the interval $[-0.025, 0.025]$

$$AR(S,T) = \frac{R(S,T) - \mathbb{E}[R(S,T)]}{1 - \mathbb{E}[R(S,T)]},$$

where R denotes the Rand Index and S and T are clusterings to compare.

4.1 Artificial data

Recall that our spatio-temporal convolution kernels possess basic properties[11] that are not shared with the baseline methods. Most notably these are invariance to permutations of the objects and to differences in speed as well as sensitivity to the spatial distribution of the objects and the direction of the movement.

In this section these properties are experimentally confirmed using two toy data sets consisting of artificially generated multi-object trajectories. The data is generated to cover a broad range of trajectories present in real-world applications. The first setting covers linear movements and each trajectory consists of five objects performing a linear movement as shown in Fig. 1 (top). To find the correct clusters, the respective kernels need to distinguish trajectories based on the direction of the movement or based on the distribution of the objects, while the objects' centroid is the same in both trajectories.

The second setting deals with circular movements and each trajectory consists of four objects moving in circles of different radii, see Fig. 1 (bottom). The construction of the

[11] The following properties refer to the $STCK_{Dist}$.

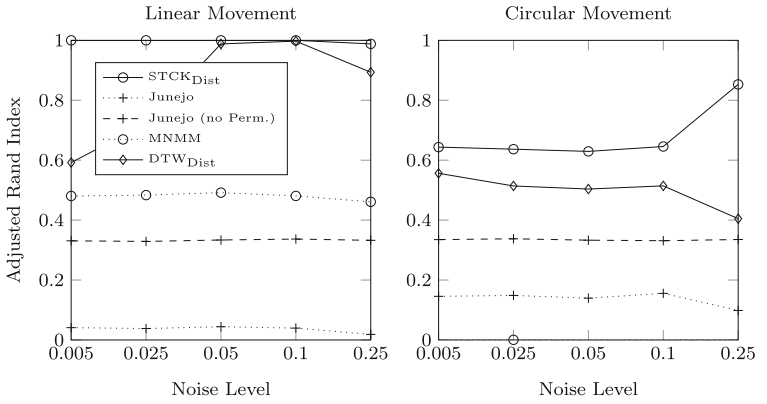

Fig. 2 Cluster performance of the STCK compared to three baselines

second set enables us to evaluate the ability of the kernels to identify the correct clusters when these only differ in the direction of the movement but a spatial separation between the clusters is not possible. Compared to linear movements, the circular task is more difficult, as the directions of only some objects differ between the clusters. The data generation is described in detail in "Appendix" section.

To evaluate the sensitivity of the methods to permutations, the objects' ordering inside each multi-object trajectory is permuted randomly in both data sets. To assess the sensitivity to changes in speed, we flip a coin for each trajectory in both test sets. With 50 % probability only every second position is retained in the trajectory. The resulting trajectory corresponds to a movement with twice the speed of the original trajectory. Finally, we add uniformly distributed noise in the range $[-\epsilon, \epsilon]$ for $\epsilon \in \{0.005, 0.025, 0.05, 0.1, 0.25\}$ with zero mean to every position in each multi-object trajectory in both test sets.

For both data sets, we generate 200 multi-trajectories (50 per cluster) for all five levels of noise. It is important to assess the methods' sensitivity to noise for two reasons. First, trajectories are generally highly variable and exact matches are very rare. Second, due to frequent tracking errors, real-life trajectories are also subject to a significant level of noise.

In this experiment, we focus on STCK$_{\text{Dist}}$ as it leads to more accurate clusters than STCK$_{\text{1-1}}$ and STCK$_{\text{Mean}}$ throughout all ranges of noise. Figure 2 compares STCK$_{\text{Dist}}$ to the three baselines. Our method outperforms dynamic time warping on both test sets and gives the most accurate clusterings. The multinomial mixture model performs reasonably well on linear movements but leads to inappropriate clusterings for circular ones. The poor results on the second data set are due to the absence of ordering information in the bag-of-directions representation when the objects move in a full circle. The Hausdorff distance-based Junejo et al. leads to generally inaccurate clusterings. We credit this finding to its sensitivity to permutations as well as negligence of ordering information.

We now evaluate the proposed approximation technique on both artificial data sets. The algorithm performs up to eleven iterations as the longest trajectory consists of eleven snapshots. Note that, at the end of the last iteration, the exact Gram matrix is obtained. The percental approximation proves accurate; the approximation quickly leads to the correct clustering after two iterations. Figure 3 shows that ASTCK also achieves higher consistencies with the correct clustering compared to all baseline methods from the second iteration onwards. Moreover, starting from the second iteration onwards it is always at least as good

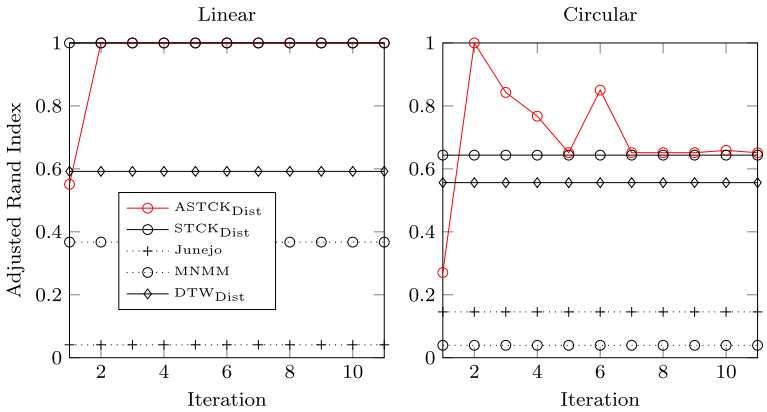

Fig. 3 Comparison of ASTCK II to the baseline methods

as the exact method. The depicted results correspond to a noise level of 0.005 but equivalent results hold for all other noise ratios.

The runtime of multi-object trajectory clustering is governed by the number of trajectories, the length of the trajectories and the number of objects. Depending on the application, usually one or two dimensions are dominating. It is thus important to evaluate the runtime for each of the three dimensions. The theoretical runtime is $O(N^2 \cdot L^2)$ spatial as well as temporal kernel evaluations, where N is the number of trajectories and L is the maximum length of the trajectories. To confirm these findings experimentally, we generate random trajectory sets in $[0, 1]^2$ and vary the number of trajectories per set, the length of the trajectories and the number of objects per trajectory. The results are depicted in Fig. 4.

The figure on top shows the results for varying numbers of trajectories. Except for multinomial mixture models that perform linearly in the number of instances, all methods exhibit a quadratic complexity, which is simply due to the fact that all pairwise similarities are computed, i.e. $N(N+1)/2$. Varying the length of the trajectories in Fig. 4 (bottom, left) shows that all STCKs as well as DTW exhibit quadratic complexities which is in line with the theoretical runtime. However, the percental approximation algorithm (two iterations) significantly improves the runtime of STCKs and is comparable to that of Junejo et al. and multinomial mixture models.

Finally, Fig. 4 (bottom, right) varies the number of objects. The results are virtually constant time complexities for all methods except STCK$_{1\text{-}1}$ and MNMM, which exhibit linear complexities. The observation is in line with our expectations as the Gaussian RBF kernel compares each object with its counterpart. The deviations in the case of STCK$_{\text{Dist}}$ for a small number of objects (less or equal two objects) is due to the additional time needed by the shrinking schemes to restore non-singularity of the covariance matrices.

4.2 Real-world data

We now evaluate spatio-temporal convolution kernels on real world data using positional data streams of ten soccer games of the German Bundesliga. The goal is to identify movement patterns by analysing the tracking data.

The tracking data is captured by the VIS.TRACK Impire (2014) tracking system during five games of Bundesliga Team A and five games of Bundesliga Team B from the 2011/12

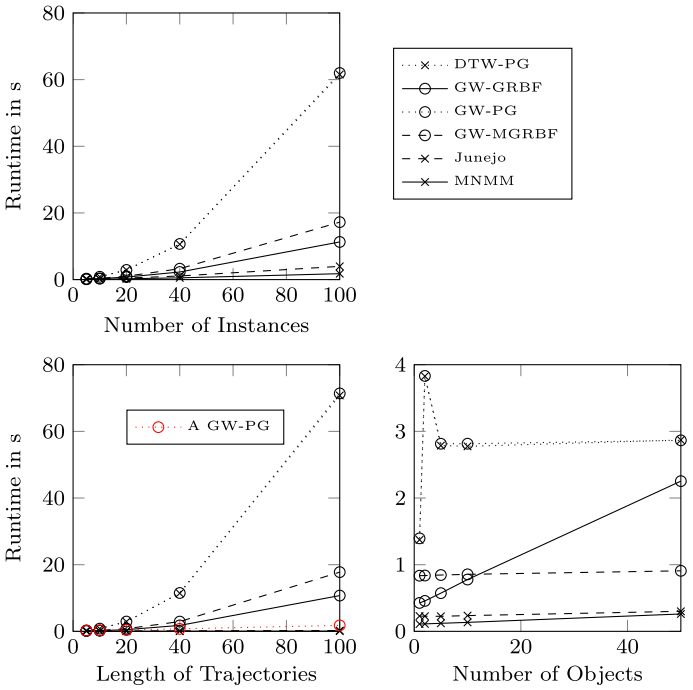

Fig. 4 Runtime: number of trajectories (*top*), length of trajectories (*bottom*, *left*), and number of objects (*bottom*, *right*)

Bundesliga season.[12] VIS.TRACK is a video based tracking system consisting of several cameras in the soccer stadium. It determines the positions of the players, ball and referees at 25 frames per second, which amounts to roughly 135, 000 positions per object and match and a total of 31, 000, 000 positions. Additionally, a marker indicates the status of the ball (in-play, stoppage) and the possession of the ball. The range of the x- (parallel to sidelines) and y-coordinate (parallel to endlines) is $[-1, 1]$, whereas values with an absolute value greater than one occur if the ball is out of bounds. The data stream is preprocessed so that positions of the second half are mirrored to account for the changeover at half time and the frame numbers of the second half are changed to succeed those of the first half. Additionally, the playing direction is determined and normalised, so that the team of interest always plays from left to right. Subsequently, we extract two types of sequences: *game initiations* and *scoring opportunities*.

Game initiations (GI) begin with the goal keeper passing the ball and end with the team loosing possession, a stoppage, the ball being in the attacking third of the field or the start of the next game initiation as defined above. Sequences shorter than length 12 are excluded. Scoring opportunities (SO) terminate when the ball is carried into a predefined zone of danger, usually defined as the attacking quarter. Scoring opportunities begin at the time of the last stoppage or win of the ball and last until the ball reaches the attacking quarter of the field $[0.5, 1] \times [-1, 1]$. Again, sequences with a length below 12 are discarded.

For every sequence there are 23 possible trajectories (ball, 22 players, no referees) to include into the analysis. Since the opposing team changes from game to game, we will

[12] The team names must not be disclosed.

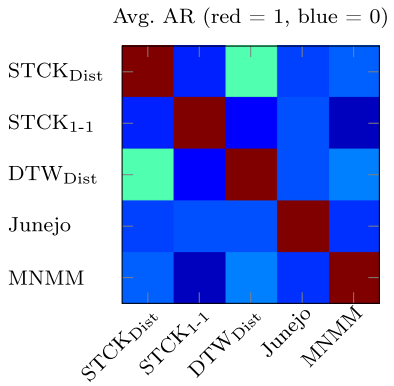

Fig. 5 Pairwise adjusted Rand indices

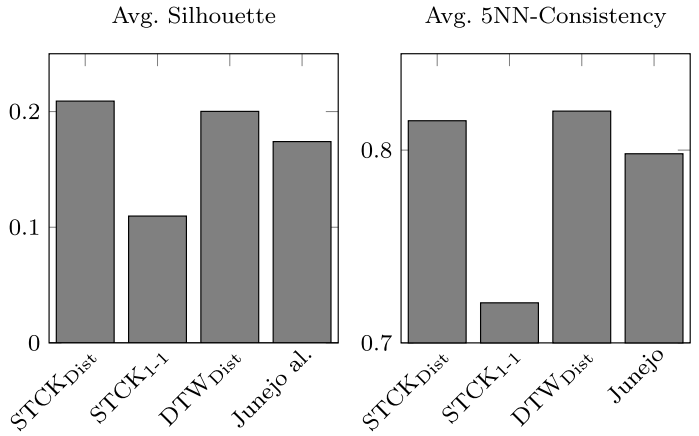

Fig. 6 Cluster consistencies and compactness

restrict the analysis to twelve objects, namely the ball and the players of Team A, and Team B respectively. In the following experiments we consider the ball as one group in the sense of Definition 2. We further include the back four (game initiations) and the four most offensive players (scoring opportunities) as a second group (in the sense of Definition 2) into the analysis. The clusterings in the remainder are thus based on the trajectories of five objects (ball, four players).

In general, there is no ground-truth for real-world clustering problems. Figure 5 therefore depicts the adjusted Rand indices between the five methods for pair-wise comparison. Generally, the adjusted Rand index is low in the majority of the cases indicating low consistency between the methods. On the other hand, we have a higher consistency between STCK$_{\text{dist}}$ and DTW$_{\text{dist}}$ demonstrating that our method is capable of dealing with speed and length differences in a similar way as dynamic time warping. This is also partially due to the use of the same kernel in both methods, once as spatial kernel and once as local distance measure.

Figure 6 (left) shows average Silhouette measures of the four similarity-based methods over the four datasets. Our STCK$_{\text{dist}}$ achieves the highest score indicating higher cluster sep-

aration and/or compactness. The object-wise STCK provides the poorest results on average, indicating that permutations are relevant in this application and that the distribution-based representation of the probability product kernel is more successful in capturing the relevant player movements. Dynamic time warping performs second best on average and in line with the previous results on the artificial datasets. Also note the high consistency with our method in Fig. 5. In terms of 5-nearest-neighbour consistency, the two distribution-based methods, $STCK_{dist}$ and DTW_{dist}, perform best, as depicted in Fig. 6 (right). Cluster quality is generally higher for game initiations compared to scoring opportunities which also results in in more interpretable clusters for these settings as we will show in the remainder.

Figures 7 and 8 illustrate the resulting clusters for the five methods. The medoids of the clusters for both teams are depicted along with the distribution of the trajectory length for each cluster. For all settings cluster membership correlates with sequence length. Generally, the cluster medoids of $\text{STCK}_{\text{dist}}$ and DTW_{dist} often coincide and are easily interpretable, while the cluster medoids of $\text{STCK}_{\text{1-1}}$ and the cluster representatives of the multinomial mixture model are difficult to make sense of.

In the Bundesliga 2011/2012 season, the strategy of Team A generally consisted of transporting the ball with few, but rehearsed short game initiations to the opposing half. For this purpose, many ball contacts were allowed and different players were integrated. On the contrary, Team B showed a rather chaotic game organisation with rather random actions and increasingly long, straight balls. Figure 7 shows the outcomes of the k-medoids cluster for the game initiations. Compared to all other methods, the the $\text{STCK}_{\text{dist}}$ capture the characteristic traits of the teams well. Team A clearly acted with many short moves (long trajectories in cluster 1) and integrated many players in the playmaking (cluttered medians). By contrast, Team B acted with many long moves (short trajectories in all clusters) and preferred linear actions.

Near the opposing goal, Team A aimed at quickly achieving a goal in the opposing half during the 2011/2012 season. They operated with only a few ball contacts and aimed to quickly transport the ball in the predefined zone of danger. Again in contrast to this, Team B had many ball contacts and took their time in waiting for a mistake of the opponent and only then played in the zone of danger to achieve a goal. Figure 8 shows that all methods are capable of retracing the different offensive strategies of both teams. Again, Team B has more solution categories (more than 33 %) than Team A, which is shown by the versatile and multifaceted running patterns. However, solely the results with $\text{STCK}_{\text{dist}}$ show that Team A rapidly tries to enter the zone of danger with very few ball contacts (short sequences in all clusters compared to Team B). To sum up, the results with $\text{STCK}_{\text{dist}}$ best reflect the game philosophies of Team A and B from a sport-scientific perspectives and are the most easy to interpret.

5 Conclusion

We presented spatio-temporal convolution kernels for multi-object scenarios. Our kernels consist of a temporal and a spatial component that can be chosen according to the characteristic traits of a problem at-hand. The computation time is quadratic in terms of the number and lengths of trajectories. We proposed an efficient percental approximation algorithm that significantly reduced the complexity to superlinear runtime. Empirical results on artificial clustering tasks showed that our spatio-temporal convolution kernels effectively identify the target concepts. Results on large-scale real world data from soccer games showed that our kernels lead to easily interpretable clusters that may be used in further analysis by coaches.

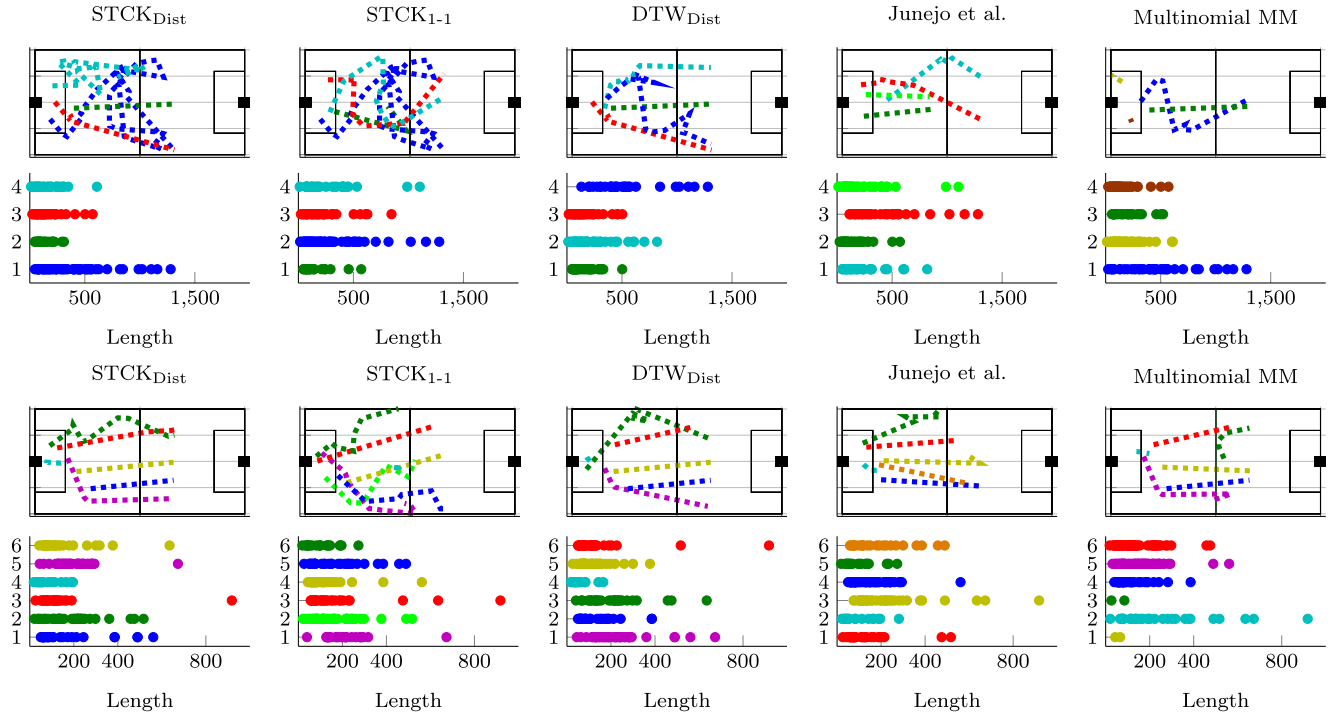

Fig. 7 Game initiation clusters of Team A (*top*; four clusters) and Team B (*bottom*; 6). The *above plots* show cluster medoids indicated by *colours*. Only the trajectory of the ball is shown, while the clusters are also based on the trajectories of four players. Where possible, similar clusters are coloured equally. Below, the distribution of the length of the trajectories in each cluster is depicted. For multinomial mixture models, the multi-object trajectory with the highest probability is depicted as representative

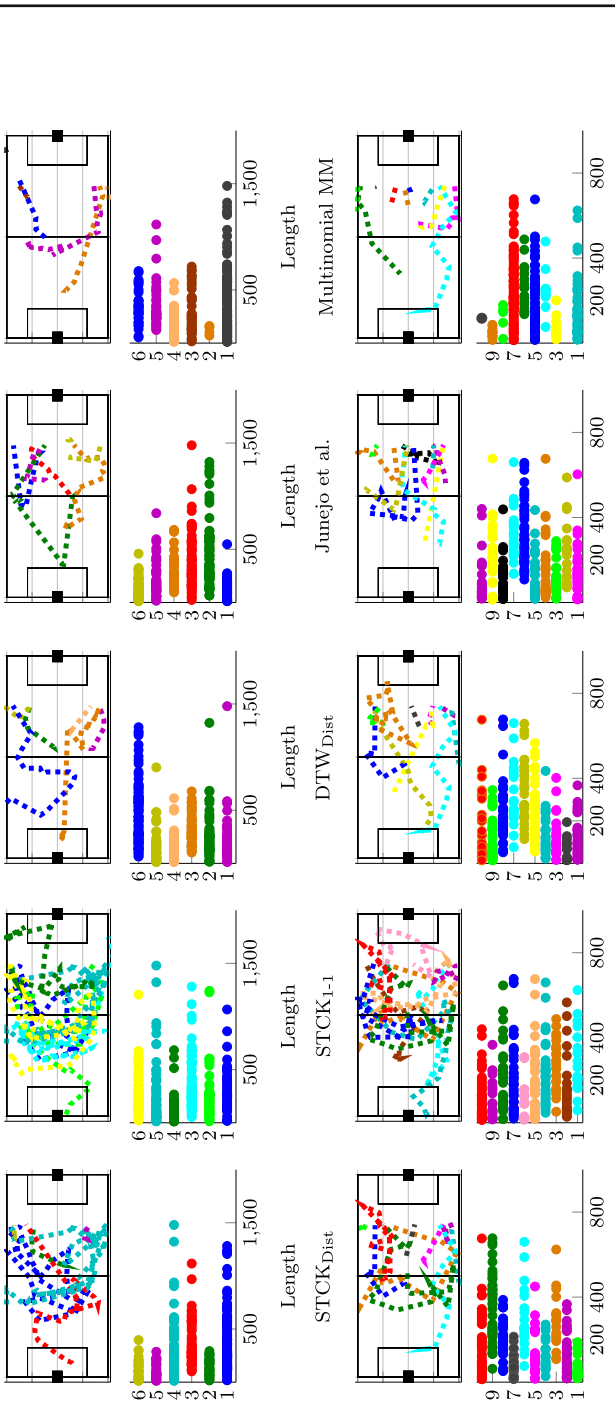

Fig. 8 Scoring opportunity clusters of Team A (*top*; six clusters) and Team B (*bottom*; 10). The plots show the cluster medoids. *Colours* indicate clusters. Only the trajectory of the ball is shown, while the clusters are also based on the trajectories of four players. Where possible, similar clusters are coloured equally. Below, the distribution of the length of the trajectories in each cluster is depicted. In the case of the multinomial mixture model, the multi-object trajectory with the highest probability is depicted as representative

Appendix: Generation of artificial data

The four clusters of the linear movement data consist of trajectories each of which has of five objects performing a linear movement as depicted in Fig. 1. The clusters differ with regard to the direction of the movement as follows:

- Cluster 1: Objects move downwards in parallel. In particular, the position (without noise) of object o_i for $i = 1, \ldots, 5$ at time $t = 0, 0.1, \ldots, 1$ is given by

$$\left(-0.2 + \frac{i-1}{10}, 1 - t\right).$$

- Cluster 2: Objects move upwards in parallel. In particular, the position of object o_i for $i = 1, \ldots, 5$ at time $t = 0, 0.1, \ldots, 1$ is given by

$$\left(-0.2 + \frac{i-1}{10}, t\right).$$

- Cluster 3: Objects move downwards while drifting apart. In particular, the position of object o_i for $i = 1, 2$ at time $t = 0, 0.1, \ldots, 1$ is

$$(-0.6 + (i-1) \cdot 0.1 + 0.4 \cdot t, 1 - t),$$

  the position of object o_3 is

$$(0, 1 - t)$$

  at time $t = 0, 0.1, \ldots, 1$ and the position of object o_i for $i = 4, 5$ at time $t = 0, 0.1, \ldots, 1$ is

$$(0.5 + (i-4) \cdot 0.1 - 0.4 \cdot t, 1 - t).$$

- Cluster 4: Objects move upwards while drifting apart.In particular, the position of object o_i for $i = 1, 2$ at time $t = 0, 0.1, \ldots, 1$ is

$$(-0.6 + (i-1) \cdot 0.1 + 0.4 \cdot (1-t), t),$$

  the position of object o_3 is

$$(0, t)$$

  at time $t = 0, 0.1, \ldots, 1$ and the position of object o_i for $i = 4, 5$ at time $t = 0, 0.1, \ldots, 1$ is

$$(0.5 + (i-4) \cdot 0.1 - 0.4 \cdot (1-t), t).$$

The second artificial data set testing on circular movements contains trajectories of four objects moving in circles of different radii. The clusters differ with regard to the orientation of the movement as follows:

- Cluster 1: All objects move clockwise. In particular, the position of object o_i for $i = 1, \ldots, 4$ at time $t = 0, 0.1, \ldots, 1$ is given by

$$(0.25i \cos(2\pi t), 0.25i \sin(2\pi t)).$$

- Cluster 2: All objects move counter-clockwise. In particular, the position of object o_i for $i = 1, \ldots, 4$ at time $t = 0, 0.1, \ldots, 1$ is given by

$$(0.25i, \cos(2\pi t), -0.25i \sin(2\pi t)).$$

– Cluster 3: The inner two objects move clockwise, the outer two counter-clockwise. In particular, the position of object o_i for $i = 1, \ldots, 4$ at time $t = 0, 0.1, \ldots, 1$ is given by

$$(0.25i, \cos(2\pi t), \text{sgn}(2 - i)0.25i \sin(2\pi t)).$$

– Cluster 4: The inner two objects move counter-clockwise, the outer two clockwise. In particular, the position of object o_i for $i = 1, \ldots, 4$ at time $t = 0, 0.1, \ldots, 1$ is given by

$$(0.25i, \cos(2\pi t), \text{sgn}(i - 3)0.25i \sin(2\pi t)).$$

References

Andrieu, C., de Freitas, N., Doucet, A., & Jordan, M. I. (2003). An introduction to MCMC for machine learning. *Machine Learning*, *50*(1–2), 5–43.

Basharat, A., Gritai, A., & Shah, M. (2008). Learning object motion patterns for anomaly detection and improved object detection. In *IEEE conference on computer vision and pattern recognition*.

Baud, O., El-Bied, Y., Honore, N., & Taupin, O. (2007). Trajectory comparison for civil aircraft. In *Aerospace conference, 2007 IEEE*.

Bellman, R., & Kalaba, R. (1959). On adaptive control processes. *IRE Transactions on Automatic Control*, *4*(2), 1–9.

Bialkowski, A., Lucey, P., Carr, P., Denman, S., Matthews, I., & Sridharan, S. (2013). Recognising team activities from noisy data. In *IEEE conference on computer vision and pattern recognition workshops*.

Bialkowski, A., Lucey, P., Carr, P., Yue, Y., & Matthews, I. (2014). "Win at home and draw away": Automatic formation analysis highlighting the differences in home and away team behaviors. In *MIT sloan sports analytics conference*.

Blanchard, G., Bousquet, O., & Massart, P. (2008). Statistical performance of support vector machines. *The Annals of Statistics*, *36*(2), 489–531.

Blei, D. M., Ng, A. Y., & Jordan, M. I. (2003). Latent dirichlet allocation. *Journal for Machine Learning Research*, *3*, 993–1022.

Buzan, D., Sclaroff, S., & Kollios, G. (2004). Extraction and clustering of motion trajectories in video. In *International conference on pattern recognition, (Vol. 2)*.

Chen, Y., Wiesel, A., Eldar, Y., & Hero, A. (2010). Shrinkage algorithms for MMSE covariance estimation. *IEEE Transactions on Signal Processing*, *58*(10), 5016–5029.

Direkoglu, C., & O'Connor, N. (2012). Team behavior analysis in sports using the poisson equation. In *IEEE international conference on image processing*.

Fowlkes, C., Belongie, S., Chung, F., & Malik, J. (2004). Spectral grouping using the Nyström method. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *26*(2), 214–225.

Fu, Z., Hu, W., & Tan, T. (2005). Similarity based vehicle trajectory clustering and anomaly detection. In *International conference on image processing, (Vol. 2)*.

Giannotti, F., Nanni, M., Pinelli, F., & Pedreschi, D. (2007). Trajectory pattern mining. In *Proceedings of the 13th ACM SIGKDD international conference on knowledge discovery and data mining, KDD '07*. ACM, New York, NY, USA. doi:10.1145/1281192.1281230.

Grunz, A., Memmert, D., & Perl, J. (2012). Tactical pattern recognition in soccer games by means of special self-organizing maps. *Human Movement Science*, *31*(2), 334–343.

Hartigan, J. A. (1975). *Clustering algorithms* (99th ed.). New York, NY: Wiley.

Hausdorff, F. (1962). *Set theory*. Chelsea: Chelsea Pub. Co.

Haussler, D. (1999). Convolution kernels on discrete structures. In *Technical report UCS-CRL-99-10, University of California at Santa Cruz*.

Hervieu, A., & Bouthemy, P. (2010). Understanding sports video using players trajectories. In *Intelligent video event analysis and understanding, studies in computational intelligence, (Vol. 332)*. Berlin, Heidelberg: Springer.

Hirano, S., & Tsumoto, S. (2005). Grouping of soccer game records by multiscale comparison technique and rough clustering. In *International conference on hybrid intelligent systems*.

Hu, W., Xiao, X., Fu, Z., Xie, D., Tan, T., & Maybank, S. (2006). A system for learning statistical motion patterns. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *28*(9), 1450–1464.

Hubert, L., & Arabie, P. (1985). Comparing partitions. *Journal of Classification*, *2*(1), 193–218.

Impire AG (2014) *VIS.TRACK Produktinformation*. http://www.bundesliga-datenbank.de/fileadmin/impire/products/Produktinformationen/VIS.TRACK_-_Produktionformation.pdf. Accessed June 10, 2014.

Intille, S. S., & Bobick, A. F. (2001). Recognizing planned multiperson action. *Computer Vision and Image Understanding*, *81*(3), 414–445.

Jebara, T., Kondor, R., & Howard, A. (2004). Probability product kernels. *Journal for Machine Learning Research*, *5*, 819–844.

Jeong, H., Chang, H. J., & Choi, J. Y. (2011). Modeling of moving object trajectory by spatio-temporal learning for abnormal behavior detection. In *IEEE international conference on advanced video and signal-based surveillance*.

Jinyang, D., Rangding, W., Liangxu, L., & Jiatao, S. (2011). Clustering of trajectories based on hausdorff distance. In *International conference on electronics, communications and control*.

Junejo, I., Javed, O., & Shah, M. (2004). Multi feature path modeling for video surveillance. In *International conference on pattern recognition, (Vol. 2)*.

Kang, C.H., Hwang, J.R., & Li, K.J. (2006). Trajectory analysis for soccer players. In *IEEE international conference on data mining workshops*.

Kang, J., & Yong, H. S. (2008). Spatio-temporal discretization for sequential pattern mining. In *Proceedings of the 2nd international conference on ubiquitous information management and communication*. ACM, New York, NY, USA.

Kang, S. J., Kim, Y., Park, T., & Kim, C. H. (2013). Automatic player behavior analysis system using trajectory data in a massive multiplayer online game. *Multimedia Tools and Applications*. doi:10.1007/s11042-012-1052-x.

Kaufman, L., & Rousseeuw, P. J. (1987). Clustering by means of medoids. In Y. Dodge (Ed.), *Statistical Data Analysis Based on the L_1-Norm and Related Methods* (pp. 405–416). North-Holland.

Kempe, M., Grunz, A., & Memmert, D. (2014). Detecting tactical patterns in basketball: Comparison of merge self-organising maps and dynamic controlled neural networks. *European Journal of Sport Science*, *15*(4), 249–255.

Kim, K., Grundmann, M., Shamir, A., Matthews, I., Hodgins, J., & Essa, I. (2010). Motion fields to predict play evolution in dynamic sport scenes. In *IEEE conference on computer vision and pattern recognition*.

Larson, J.S., Bradlow, E.T., & Fader, P.S. (2005). An exploratory look at supermarket shopping paths. *International Journal of Research in Marketing, 22*. doi:10.1016/j.ijresmar.2005.09.005.

Lazebnik, S., Schmid, C., & Ponce, J. (2006). Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *IEEE computer society conference on computer vision and pattern recognition, (Vol. 2)*.

Ledoit, O., & Wolf, M. (2004). A well-conditioned estimator for large-dimensional covariance matrices. *Journal of Multivariate Analysis*, *88*(2), 365–411.

Lee, J.G., Han, J., & Li, X. (2008). Trajectory outlier detection: A partition-and-detect framework. In *Proceedings of the 2008 IEEE 24th international conference on data engineering*. IEEE Computer Society: Washington, DC, USA. doi:10.1109/ICDE.2008.4497422.

Li, R., & Chellappa, R. (2010). Group motion segmentation using a spatio-temporal driving force model. In *IEEE conference on computer vision and pattern recognition*.

Li, R., Chellappa, R., & Zhou, S. (2009). Learning multi-modal densities on discriminative temporal interaction manifold for group activity recognition. In *IEEE conference on computer vision and pattern recognition*.

Lin, C. J. (2001). On the convergence of the decomposition method for support vector machines. *IEEE Transactions on Neural Networks*, *12*(6), 1288–1298.

Lin, D., Grimson, E., & Fisher, J. (2009). Learning visual flows: A lie algebraic approach. In *IEEE conference on computer vision and pattern recognition*.

Lucey, P. J., Bialkowski, A., Carr, P., Morgan, S., Matthews, I., & Sheikh, Y. (2013). Representing and discovering adversarial team behaviors using player roles. In *IEEE conference on computer vision and pattern recognition*.

Mamoulis, N., Cao, H., Kollios, G., Hadjieleftheriou, M., Tao, Y., & Cheung, D.W. (2004). Mining, indexing, and querying historical spatiotemporal data. In *Proceedings of the tenth ACM SIGKDD international conference on knowledge discovery and data mining*. ACM: New York, NY, USA.

Mehta, S., Parthasarathy, S., & Yang, H. (2005). Toward unsupervised correlation preserving discretization. *IEEE Transactions on Knowledge and Data Engineering*, *17*(9), 1174–1185.

Ng, A. Y., Jordan, M. I., & Weiss,Y. (2001). On spectral clustering: Analysis and an algorithm. In *Advances in neural information processing systems*, pp. 849–856.

Ong, C. S., Canu, S., Smola, A. J. (2004). Learning with non-positive kernels. In *International conference on machine learning*.

Pao, H. K., Chen, K. T., & Chang, H. C. (2010). Game bot detection via avatar trajectory analysis. *IEEE Transactions on Computational Intelligence and AI in Games*, *2*(3), 162–175.

Perše, M., Kristan, M., Kovačič, S., Vučkovič, G., & Perš, J. (2009). A trajectory-based analysis of coordinated team activity in a basketball game. *Computer Vision and Image Understanding*, *113*(5), 612–621.

Piciarelli, C., Foresti, G., & Snidaro, L. (2005). Trajectory clustering and its applications for video surveillance. In *IEEE conference on advanced video and signal based surveillance*.

Rand, W. M. (1971). Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical Association*, *66*(336), 846–850.

Rousseeuw, P. J. (1987). Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics, 20*.

Saleemi, I., Shafique, K., & Shah, M. (2009). Probabilistic modeling of scene dynamics for applications in visual surveillance. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 31*(8).

Shawe-Taylor, J., & Cristianini, N. (2004). *Kernel methods for pattern analysis*. Cambridge: Cambridge University Press.

Siddiquie, B., Yacoob, Y., & Davis, L. (2009). *Recognizing plays in american football videos*.

Steinwart, I. (2002). Support vector machines are universally consistent. *Journal of Complexity*, *18*(3), 768–791.

Trunfio, G. A., D'Ambrosio, D., Rongo, R., Spataro, W., & Di Gregorio, S. (2011). A new algorithm for simulating wildfire spread through cellular automata. In *ACM transactions on modeling and computer simulation*.

Vapnik, V. N. (1995). *The nature of statistical learning theory*. New York: Springer.

Wang, X., Ma, K. T., Ng, G. W., & Grimson, W. (2008). Trajectory analysis and semantic region modeling using a nonparametric bayesian model. In *IEEE conference on computer vision and pattern recognition*.

Wauthier, F. L., Jojic, N., Jordan, M. I. (2012). Active spectral clustering via iterative uncertainty reduction. In *ACM SIGKDD international conference on knowledge discovery and data mining*. ACM.

Wei, X., Sha, L., Lucey, P., Morgan, S., & Sridharan, S. (2013). Large-scale analysis of formations in soccer. In *International conference on digital image computing: Techniques and applications*.

Williams, C., & Seeger, M. (2001). Using the Nyström method to speed up kernel machines. In *Advances in neural information processing systems 13* (pp. 682–688). MIT Press.

Zhu, G., Huang, Q., Xu, C., Rui, Y., Jiang, S., Gao, W., & Yao, H. (2007). Trajectory based event tactics analysis in broadcast sports video. In *International conference on multimedia*. ACM.